Proceedings of theChelyabinsk Scientific Center, No 4(9), 2000

MATHEMATICS

# THE ALGORITHM FOR TRACING OF FLAT EULER CYCLES WITH ORDERED ENCLOSING

**T. A. Panioukova (1), A. V. Panyukov (2)**

E-mail: kwark@mail.ru (1); panyukov@inf.susu.ac.ru (2)

Southern Ural State University, Chelyabinsk, Russia

Received: November 20, 2000

## Flat Euler cycles with ordered enclosing

Let on a plane $S$ be drawn flat Euler graph $G \subset S$ ([1], [2]) with set of vertexes $VG \subset S$, set of edges $EG \subset S$ and set of faces $FG \subset S$, and let $f_0 \subset S$ be external face of this graph. Let $\mathrm{Int}\,(H)$ be interior of subset $H \subset G$, i.e. join of all connected components of set $S \setminus H$ not containing external face $f_0$. Further let us designate the number of elements of set $M$ as $|M|$.

**Definition 1** *Let us name Euler cycle $C = v_1e_1v_2e_2 \dots e_{|EG|}v_1$ with labelled vertex $v_1$ as a cycle with $v$-ordered enclosing if for any its initial part $C_l = v_1e_1v_2 \dots e_l$, $l \leq (|EG|)$ it takes place that $\mathrm{Int}(C_l) \bigcap EG = \emptyset$, i.e. the intersection of $C_l$ interior with set of edges is empty.*

For flat Euler graph on Fig. 1, cycle $v_1e_1v_2e_2v_3e_3v_1e_4v_2e_5v_3e_6v_1$ satisfies the given condition of $v$-ordered enclosing and cycle $v_1e_4v_2e_5v_3e_6v_1e_1v_2e_2v_3e_3v_1$ does not satisfy because $\mathrm{Int}\,(v_1e_4v_2e_5v_3e_6v_1) \supset \{e_1, e_2, e_3\}$.

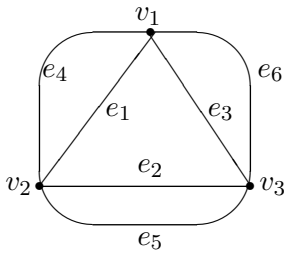


Fig. 1. Graph for illustration of Euler cycles

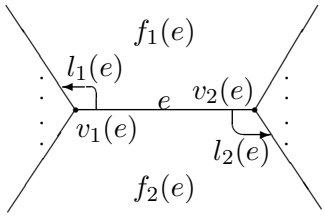


Fig. 2. The functions on a set of edges of a graph

The used definitions are interpreted in terms of cutting problem as following: $S$ is cutting out sheet, $G$ is cutting out plan, $C$ is trajectory of cutting tool movement, $\mathrm{Int}(C_l)$ is part cut off from a sheet at passage by cutting tool of a part of trajectory. *Condition of $v$-ordered enclosing* means that part cut off from a sheet does not require additional slitings.

Correctness of the problem of tracing of flat Euler cycle with odered enclosing shows

**Theorem 1** *Let $G$ be a flat Euler graph, $v \in VG$ is vertex adjacent external face of graph $G$. Then there is the Euler cycle with $v$-ordered enclosing.*

The proof is by mathematical induction on number of faces of graph $G$. Euler graphs containing two faces are the simple cycles. There is nothing to prove for simple cycles. Let any flat Euler graph with number of faces $m: \ 2 < m < K$ has a cycle with $v$-ordered enclosing for any its vertex $v \in VG$ adjacent external face. Without loss of generality we can assume that the degrees of all vertexes of graph $G$ more or equal 4.

Let $f_0$ be external face of graph $G$, and $C(f_0) = v_1e_1v_2e_2\dots e_Lv_1$ represents a cycle from edges of graph $G$ bounding external face $f_0$. Graph $\hat{G} = G\backslash EC(f_0)$ resulted by removal of graph $G$ edges bounding face $f_0$ contains not more than $|VC(f_0)|$ connected components. Let $T$ be transversal of partition the set $VC(f_0)$ on subsets of $\hat{G}$-connected vertexes. Let $G(t)$ be conponent of connectivity of graph $\hat{G}$ containing vertex $t \in T$ adjacent external face.

It is obvious that $G(t)$, $t \in T$ are flat Euler cycles containing less then $K$ faces. Hence, each of graphs $G(t)$, $t \in T$ has an Euler cycle with $t$-ordered enclosing. The cycle for grapf $G$ can be constructed from cycle $C(f_0)$ by replacement of each vertex $t \in T$ by an Euler cycle with $t$-ordered enclosing for graph $G(t)$.

Theorem 1 is proved.

Actually recursive algorthm for tracing of Euler cycle with v-ordered enclosing is described in the proof of theorem 1. The recurcive algorithm is not rather definite. For example, transversals of a set of subsets can be selected ambiguously. Besides the complexity of each step essentially depends on a mode of coding graph $G$. Using man-machine technologies it is possible to remove this ambiguity. However, it is possible to go on a path of an improvement the algorithm, simultaniously tending to minimize the complexity of algorithm. Therefore, attempt is made for giving more formal exposition of algorithm, free from indicated singularities.

## Algorithm

Let us set define six functions: $v_1(e)$, $v_2(e)$ are vertexes incident to edge $e \in EG$; $f_k(e)$ is the face which edge $e$ passes during its rotation counter-clockwise around the vertex $v_k(e)$, where $k = 1, 2$; $l_k(e)$ is the edge belonging the frontier of fecet $f_k(e)$ and incident vertex $v_k(e)$, $k = 1, 2$. Introduced functions $v_k() : EG \to VG$, $l_k() : EG \to EG$, $f_k() : EG \to FG$, $k = 1, 2$ are shown on fig.2. The constructing of these functions does not make any problems. Actually they are determined during designing of graph $G$.

Text of algorithm tracing an Euler cycle with $v$-ordered enclosing in graph $G$ is shown in fig. 3.

```
begin
  input:
    v_k() : EG → VG,  k = 1,2,
    l_k() : EG → EG,  k = 1,2,
    f_k() : EG → FG,  k = 1,2,
    f_0 ∈ FG
  output:
    first ∈ EG,    last ∈ EG,
    mark() :  EG → EG

  procedure REPLACE(e ∈ EG)
  begin
    rv := v_2(e);  rl := l_2(e);
    rf := f_2(e);  v_2(e) := v_1(e);
    l_2(e) := l_1(e);  f_2(e) := f_1(e);
    v_1(e) := rv;  l_1(e) := lv;  f_1(e) := rf;
    return;
  end of REPLACE

  Initialization:
  (∀v ∈ VG) do S(v) := ∅ od;
    (∀e ∈ EG) do
      mark(e) = ∞;
      if ((f_1(e) = f_0) ∨ (f_2(e) = f_0)) e_0 := e;
      od
    if (f_2(e_0) = f_0) REPLACE(e_0);
    first := last := e_0;
    v_0 := v := v_1(e_0); ne := l_1(e_0);
    /*   k := 1;   kmark(e_0) := 1;  */

Ordering:
  while(first ≠ ∞) do
    while(mark(ne) = ∞) do
      M1:/*  kmark(ne) := k;  */
      mark(last) := ne;
      if (v_2(ne) ≠ v)REPLACE(ne);
      v := v_1(ne); ne := l_1(ne);
      od
    e := first; first := mark(first);
    v := v_2(e); ne := l_2(e);
    M2:/*  k = kmark(e) + 1;  */
    mark(e) := S(v);  S(v) := e;
    od

  Forming:
    v := v_0; e := S(v); first := last := e;
    while(last ≠ ∅) do
      S(v) := mark(e);
      v := v_1(e);  e := S(v);
      M3: mark(last) := e; last := e;
      od
    stop;

    end.
```

Fig.3. Text of the algorithm

Input data of the algorithm are defined above functions and pointer $f_0$ to exterior face. Output data of the algorithm are the edge pointers $first$ and $last$, and array *mark*. This array is the representation of function $EG \to EG$, $mark(e)$ is edge from the constructed Euler cycle following $e$.

The realization of algorithm can be divided into three stages: 1) "Initialization"; 2) "Ordering"; 3) "Forming". Beginning of each stage is marked by the corresponding comment. In a skew field of algorithm the array *mark* is redefined for three times: at queueing of $M1$-marked edges (Fig. 4) by operator $M1$; at constructing of stacks of vertexes (Fig. 5) by operator $M2$; and at defining of the final value (Fig. 6) by operator $M3$.

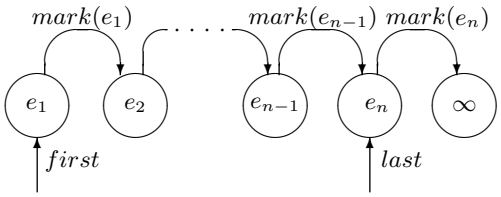


Fig. 4. Organization of queue of M1-marked edges

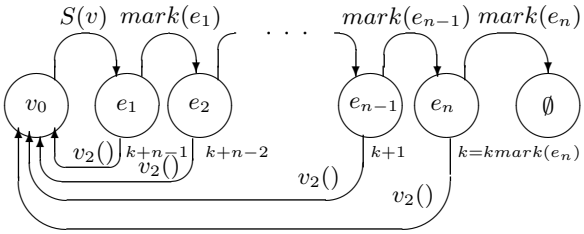


Fig. 5. Organization of stacks of M2-marked edges

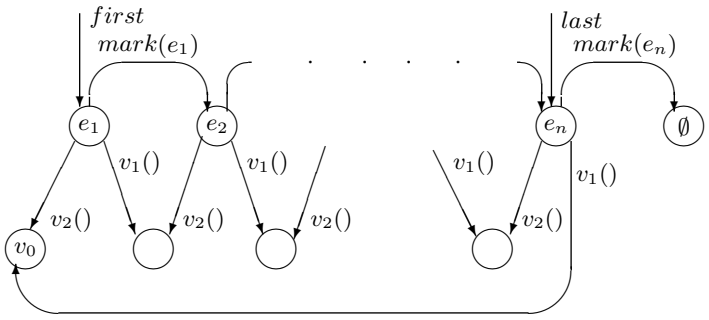


Fig. 6. Organization of edge tour with ordered enclosing in the Euler cycle

On stages "Initialization" and "Ordering" algorithm uses procedure *REPLACE* for redefinition of the functions $v_k()$, $l_k()$ and $f_k()$ so that the moving along edge $e$ happens from vertex $v_2(e)$ to $v_1(e)$. Obviously, this redefinition is exchange of index $k$ to $3-k$ for functions $v_k()$, $l_k()$, $f_k()$, $k=1,2$. While describing and analyzing the algorithm let us use $\bar{v}_k()$, $\bar{l}_k()$, $\bar{f}_k()$, $k=1,2$ for functions constructed by the algorithm despite the predefined functions $v_k()$, $l_k()$, $f_k()$, $k=1,2$.

At *Initialization* stage the initial values of all the variables are appropriated: 1) all stacks are declared empty $S(v)=\emptyset, \quad v \in VG$; all edges are unmarked $mark(e)=\infty$; 2) first edge $e \in EG$ (any of edges bounding exterior face $f_0$) is determined; 3) if the function $f_2(e)$ coincides with an exterior face procedure *REPLACE* is used;

The queue of $M1$-marked edges is initialized as consisting from initial edge $e_0$, variables *first* and *last* that are used for pointing on first and last elements of queue accordingly point on edge $e_0$. The variable $ne$ is used for the definition of the next edge, which is included in the list of $M1$. The variable $v_0$ is used for saving the vertex adjacent to the exterior face, and variable $v$ is used as current vertex for definition the orientation of edge $ne$.

The *Ordering* stage is fulfilled as follows. First of all all edges belonging to connected component, and bounding face $f_0$ are included in the list of $M1$. Procedure *REPLACE* is used for making the indexing of vertexes met to an index. After that, edge $e$ which has been included in $M1$-queue at $k$ loop is excluded from this queue on $k+1$ loop and pushed into the stack of vertex $v_2(e)$ (i.e. the status of edge $e$ is becoming $M2$-marked). All unmarked edges bounding faces common with ones deleted from $M1$-queue in this loop are included into queue of $M1$-marked edges.

The variable $k$ is used for analisys of algorithm productiveness. This variable is used as counter of stages. Also we determine the function $kmark() : EG \to$ N that shows the number of stage when an edge is put into the queue of $M1$- marked edges. In the text of algorithm (Fig.3 the operators that determine values of $k$ and $kmark()$ are in comments $/*\ldots*/$.

Let us introduce

$$E_k = \{e \in EG : kmark(e) = k\}, \quad A_k = \{(\bar{v}_1(e), \bar{v}_2(e)) : e \in E\},$$
$$E_k^* = \{e \in EG : kmark(e) \le k\}, \quad A_k^* = \{(\bar{v}_1(e), \bar{v}_2(e)) : e \in E_k^*\},$$
$$\overline{E}_k = EG \backslash E_k^*.$$

Let $G(E)$ be flat graph burn by set of edges $E \subset EG$, and $G^*(A)$ be orgraph burn by set of arcs $A$.

**Lemma 1** *Let $k \le M = \max\limits_{e \in EG} kmark(e)$. Then*

*1) $\mathsf{Int}(E_k) \supset \overline{E}_k$; $S \backslash \mathsf{Int}(E_k) \supset E_k^*$;*

*2) $G(E_k)$ is the join of cycles not intersected on edges;*

*3) $G^*(A_k)$ is the oriented graph representing the join of oriented cycles not intersected on arches.*

**Lemma 2**

$$\left(M = \max_{e \in E} kmark(e)\right) \Rightarrow \left(\overline{E}_M = \emptyset\right).$$

It follows from lemma 2 that the algorithm determines the value of $kmark()$ function for each edge $e \in EG$. It means that each edge $e \in EG$ is included into the queue of M1-marked edges. Such an inclusion is available only once because when we include edge $e \in EG$ into the queue we get $mark(e) \ne \infty$. Each edge which has got into the queue of M1-marked edges can only be transfered into the state of M2-marked after its inclusion into stack of $\bar{v}_2(e)$ vertex. The order of such an inclusion is determined by a queue of M1-marked edges. Thus, after ending of "Ordering" stage for each vertex $v \in VG$ we have (see Fig. 5.)

$$S(v) = \arg \max_{e \in E(v)} kmark(e);$$

$$\left(e' \in E(v), \quad kmark(e') > \min_{e \in E(v)} kmark(e)\right) \Rightarrow \left(\begin{array}{rcl} e'' & = & mark\,(e') \in E\,(v) \\ kmark\,(e'') & = & kmark\,(e') - 1 \end{array}\right),$$

where $E(v) = \{e \in EG | v = \bar{v}_2(e)\}$. Besides it comes from lemmas 1 and 2 that

$$E_M^* = EG, \qquad M = \max_{e \in E(v)} kmark(e).$$

Therefore oriented graph $G^* = G^*(A_M^*)$ as the oriented image of graph $G$ is connected. As $G^*$ is the assosiation of the oriented cycles not intersected on arches $G^*$ is Euler graph.

According to the description "Forming" stage the algorithm constructs maximal on inclusion chain $C = v_1 e_1 v_2 e_2 v_3 \ldots e_L v_{L+1}$ which satisfies the following conditions:

a) $v_1 = v_0$ is vertex adjacent the exterior face that has been found at "Initialization" stage, and

$$e_i = \arg \max_{e \in E(v_i) \backslash \{e_l | l < i\}} kmark(e), \quad v_{i+1} = \bar{v}_1(e_i), \qquad i = 1, 2, \ldots, L;$$

b) for any beginning part $C_l = v_1 e_1 v_2 e_2 \ldots, e_l$, $l \le L$ and for any vertex $v \in VC$ it follows that

$$\min_{e \in E(v) \cap EC_l} kmark(e) > \max_{e \in E(v) \backslash EC_l} kmark(e).$$

Since the oriented graph $G^*$ is Euler we obtain $C$ is a cycle. Obviously this cycle contains all the edges incident to vertex $v_1$, and, hence, all the vertexes belonging the boundary of exterior face.

**Lemma 3** *For all $l = 1, 2, \ldots, L$ and $k = 1, 2, \ldots, M$ the equality* $\mathrm{Int}\,(C_l) \bigcap E_k = \emptyset$ *takes place.*

As $EG = \bigcup_{k=1}^{M} E_k$ then according to lemma 3 we have $\mathrm{Int}(C_l) \cap EG = \emptyset$, $l = 1, 2, \ldots, L$. Taking all the above we can conclude that $L = |EG|$ and the cycle constructed on "Forming" stage is Euler cycle with ordered enclosing.

It is obvious that computational complexity of "Initialization" and "Forming" stages is $O\,(|EG|)$. Computational complexity of "Ordering" stage is also $O\,(|EG|)$ as we see that each edge is only once put into the queue of M1-marked edges and then the vertex is removed from it to the vertexes' stacks. Computational complexity of these operations is $O(1)$. So, computational complexity of the whole algorithm is $O\,(|EG|)$.

All stated above we generalize as

**Theorem 2** *If $G = (EG, VG)$ be a flat Euler graph with set of faces $FG$, functions $v_k() : EG \to VG$, $l_k() : EG \to EG$, $f_k() : EG \to FG$, $k = 1, 2$, that are determined on $EG$, then the algorithm represented at Fig. 3 finds the Euler cycle with ordered enclosing. After stopping of the algorithm variables $first$ and $last$ determine the first and last edges of the found cycle, the value of $mark(e)$ function is edge that follows after the edge $e \in EG$ in the found cycle. Computational complexity of algorithm is the value $O\,(|EG|)$.*

## Conclusion

So, the problem of tracing of special Euler cycles in flat graph is stated. The cycle restrictions are defined by the requirements which arises during automatic projection of supervisors for cutting prosess. The existance of such cycles is proved, algorithm of their tracing is offered, and its correctness is proved.

## References

1. *Christofides N.* Graph theory. An algorithmic approach. – Academic Press Inc. (London)Ltd. 1977.

2. *Fleischner H.* Eulerian graphs and related topics. Vol. 1-2. – Amsterdam. Noth. Holl. 1991.